(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **08115357 A**

(43) Date of publication of application: **07.05.96**

(51) Int. Cl **G06F 17/50**

(21) Application number: **06253510**

(22) Date of filing: **19.10.94**

(71) Applicant: **FUJITSU LTD**

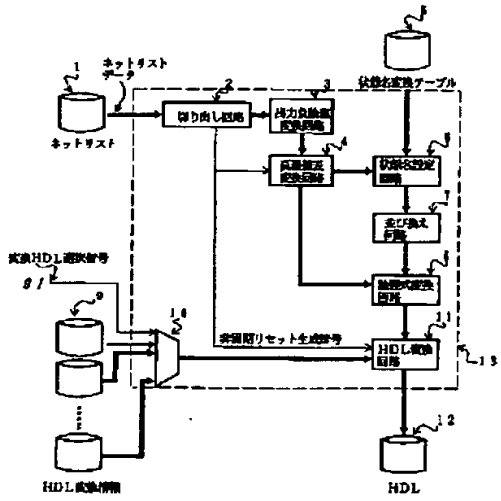
(72) Inventor: **IGUCHI KATSUMI**

(54) METHOD AND DEVICE FOR CONVERSION OF NET LIST INTO HARDWARE LANGUAGE

(57) Abstract:

PURPOSE: To convert held net list resources into the hardware description language(HDL) by converting a combinational circuit into a truth table, expanding the ignorance logic of the current state code and next state code of the truth table, and generating the HDL from the truth table containing state names.

CONSTITUTION: A truth value table converting circuit 4 converts the combinational circuit 30 after the negative logic output of a flip-flop part was removed by an output negative logic converting circuit 3 into a truth value table. Then, the ignorance logic of the current state code and next state code is expanded. In the expanded truth value table, temporary state name is assigned to a state code. A rearranging circuit 7 sorts the truth value table wherein the temporary name is set with the current state name to put codes together by the current state name. When an asynchronous reset generation signal is inputted from a segmentation circuit 2 to the HDL converting circuit 11, the HDL is generated from an asynchronous reset conditional expression on the basis of hardware conversion information corresponding to a conversion HDL select signal 91. Further, the HDL is generated from the truth value table containing the state names and the processing is ended.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－115357

(43)公開日　平成８年(1996)５月７日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ		技術表示箇所
Ｇ０６Ｆ　17/50					
		9191－5H	Ｇ０６Ｆ　15/ 60	６５４　Ａ	
		9191－5H		６５４　Ｋ	

審査請求　未請求　請求項の数２　ＯＬ　（全 ９ 頁）

(21)出願番号　特願平6－253510

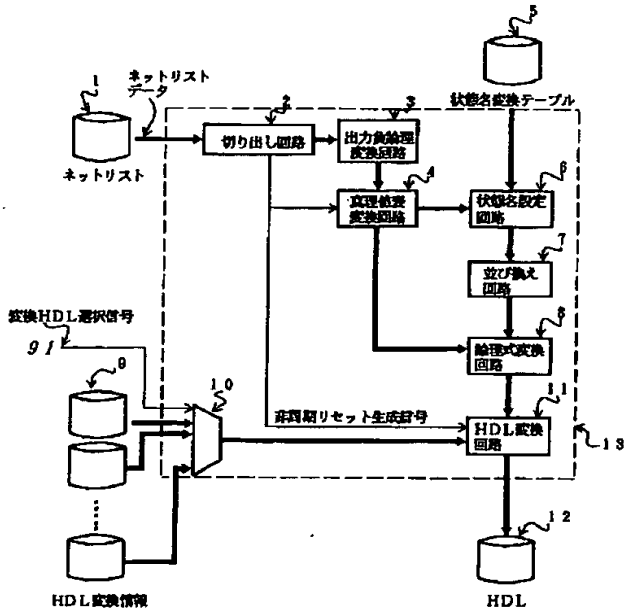
(22)出願日　平成６年(1994)10月19日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　井口　克己
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　林　恒徳

(54)【発明の名称】　ネットリストのハードウェア言語への変換方法及び装置

(57)【要約】
【目的】状態遷移回路に基づく接続情報がリストされたネットリストのハードウェア記述言語への変換方法を提供する。
【構成】フリップフロップで構成されるレジスタ部分とレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストからフリップフロップ部分と組み合わせ回路部分を切り出し、フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて組み合わせ回路を真理値表に変換し、真理値表の現状態コード、次状態コードの無視論理を展開し、現状態コード、次状態コードを仮りの状態名に置き換え、現状態名毎に該真理値表をまとめ、次いで、状態名を含んだ真理値表からハードウェア記述言語を生成する。

実施例構成ブロック図



1

【特許請求の範囲】

【請求項１】フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストから該フリップフロップ部分と組み合わせ回路部分を切り出し、

該フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて該組み合わせ回路を真理値表に変換し、

該真理値表の現状態コード、次状態コードの無視論理を展開し、

該現状態コード、次状態コードを仮りの状態名に置き換え、

現状態名毎に該真理値表をまとめ、

次いで、状態名を含んだ真理値表からハードウェア記述言語を生成するようにしたことを特徴とするネットリストのハードウェア記述言語への変換方法。

【請求項２】フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストを記憶する第一のメモリと、

状態名変換テーブルを記憶する第二のメモリと、

特定のハードウェア記述言語への変換情報を記憶する第三のメモリと、

演算処理装置を有し、

該演算処理装置は、該第一のメモリからのネットリストから該フリップフロップ部分と組み合わせ回路部分を切り出する切り出し、

該フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて該組み合わせ回路を真理値表に変換し、

該真理値表の現状態コード、次状態コードの無視論理を展開し、

該現状態コード、次状態コードを該第二のメモリからの状態名変換テーブルに基づき、仮りの状態名に置き換え、

現状態名毎に該真理値表をまとめ、

次いで、状態名を含んだ真理値表から該第三のメモリからの特定のハードウェア記述言語への変換情報に基づき、ハードウェア記述言語を生成するように構成されたことを特徴とするネットリストのハードウェア記述言語への変換装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、ネットリストのハードウェア記述言語への変換方法及び装置に関し、特に、フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストのハードウェア記述言語への変換方法及び装置に関する。

2

【０００２】

【従来の技術】従来回路設計を行うに際し回路図を直接手で書いたり、或いはＣＡＤ（Computer Aided Design）システムを用いずに、電気的特性に依存しない機能マクロを用いた機能図を使って回路設計を行う場合がある。

【０００３】即ち、設計者は、この機能図を基にして回路の機能検証を行う機能シュミレーションや実際のセルにマッピングを行う回路合成を行って、目的とする回路を設計していた。

【０００４】しかし、機能図のみでは、目的とする回路の機能を十分に表現出来ない。このために回路の動作をソフトウェアのプログラミング言語のように柔軟に記述できるようにし、回路設計用のプログラミング言語で回路を設計することが、現在の回路設計の主流になりつつある。

【０００５】一方、この回路設計用のプログラミング言語は、ハードウェア記述言語（ＨＤＬ）と呼ばれ、世界的に広く使用されているハードウェア記述言語（ＨＤＬ）として、ＶＨＤＬあるいは、Verilog-ＨＤＬと呼ばれるものがある。

【０００６】市販されているシュミレータや回路合成システムもこれらのハードウェア記述言語（ＨＤＬ）を入力とするものが多く有る。したがって、市販のＣＡＤシステムを使用する時やさまざまなＣＡＤシステムとインタフェースをとる場合には、ハードウェア記述言語（ＨＤＬ）を経由して設計データをＣＡＤシステムに入力する必要が生じる。

【０００７】しかし、かかるハードウェア記述言語（ＨＤＬ）ではない、接続情報のリストであるネットリストを資産として保有する場合は、これまでは、市販のＣＡＤシステムを使用しようとする場合、そのままでは使用出来ないことになる。

【０００８】したがって、ネットリストをハードウェア記述言語（ＨＤＬ）に変換する必要がある。この変換に際し、セルで記述されたランダムロジックは、対応する論理をそのままハードウェア記述言語（ＨＤＬ）の論理式に変換すればよい。

【０００９】しかし、フリップフロップ（ＦＦ）で構成された制御部分とランダムロジックで構成された組み合わせ回路部分を持つような状態遷移回路については、ハードウェア記述言語（ＨＤＬ）に変換する場合、割りつけられた状態コードを見つけ出す技術が必要であった。

【００１０】そして、この割りつけられた状態コードを見つけ出す技術として、これまで適当な方法がなく、したがって、保有するネットリスト資産を市販のＣＡＤシステムにおいて使用することが出来ないと言う問題が存在した。

【００１１】

【発明が解決しようとする課題】したがって、本発明の目的は、保有するネットリスト資産をハードウェア記述

言語（ＨＤＬ）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【００１２】更に、本発明の目的は、特にフリップフロップ（ＦＦ）で構成された制御部分とランダムロジックで構成された組み合わせ回路部分を持つような状態遷移回路についてのネットリスト資産をハードウェア記述言語（ＨＤＬ）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【００１３】また、本発明の目的は、ハードウェア記述言語に依存しないで、保有するネットリスト資産をハードウェア記述言語（ＨＤＬ）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【００１４】

【課題を解決するための手段及び作用】本発明にしたがうネットリストのハードウェア記述言語への変換方法は、先ずフリップフロップで構成されるレジスタ部分とこのレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストからフリップフロップ部分と組み合わせ回路部分を切り出す。

【００１５】次いで、フリップフロップ部分の負論理出力を組み合わせ回路部分から取り除いて、組み合わせ回路を真理値表に変換し、この真理値表の現状態コード、次状態コードのdon't care（無視論理）を展開し、現状態コード、次状態コードを仮りの状態名に置き換え、現状態名毎に真理値表をまとる。

【００１６】次いで、状態名を含んだ真理値表からハードウェア記述言語を生成するようにする。

【００１７】更に、本発明にしたがうネットリストのハードウェア記述言語への変換装置は、フリップフロップで構成されるレジスタ部分とこのレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストを記憶する第一のメモリと、状態名変換テーブルを記憶する第二のメモリと、特定のハードウェア記述言語への変換情報を記憶する第三のメモリと、演算処理装置を有する。

【００１８】そして、演算処理装置は、第一のメモリからのネットリストからフリップフロップ部分と組み合わせ回路部分を切り出する切り出す。フリップフロップ部分の負論理出力を組み合わせ回路部分から取り除いて、組み合わせ回路を真理値表に変換する。

【００１９】そして、真理値表の現状態コード、次状態コードのdon't care（無視論理）を展開し、現状態コード、次状態コードを該第二のメモリからの状態名変換テーブルに基づき、仮りの状態名に置き換え、現状態名毎に該真理値表をまとめる。

【００２０】次いで、状態名を含んだ真理値表から第三のメモリからの特定のハードウェア記述言語への変換情報に基づき、ハードウェア記述言語を生成するように構成される。

【００２１】

【実施例】図１は、本発明にしたがうネットリストのハードウェア記述言語への変換装置の実施例ブロック図である。更に図２は、図１に対応する実施例処理フローである。

【００２２】図１において、１は、状態遷移回路の接続情報を記述するネットリストを記憶するメモリ装置である。１３は、本発明にしたがうネットリストのハードウェア記述言語への変換方法を実行制御する装置であり、演算処理装置によって構成される。

【００２３】この変換方法を実行制御する装置１３内には、各実行制御の過程の機能を回路部分として示し以下、回路として説明するが、ハード回路部分として実現することも、或いは対応する回路機能をソフトウェア処理により実行することも可能である。

【００２４】５は、後に説明するネットリストのハードウェア記述言語への変換方法の実行の際に真理値表に状態名を設定する際の状態名を記述する状態名変換テーブルを記憶するメモリである。

【００２５】９は、複数のＨＤＬ変換情報であり、例えば既述したＶＨＤＬ、Verilog-ＨＤＬ等のハードウェア記述言語（ＨＤＬ）の変換情報を記憶するメモリである。この内、オペレータの操作により変換ＨＤＬ信号９１を入力して、変換を行うべき対応するハードウェア記述言語（ＨＤＬ）の変換情報が選択される。

【００２６】１２は、変換されたハードウェア記述言語（ＨＤＬ）を記憶するメモリ装置である。

【００２７】次に変換方法を実行制御する装置１３における変換の実際を以下に説明する。

【００２８】図２の動作フローにおいて、変換処理がスタートすると先ず図１の切り出し回路２において、記憶装置１のネットリストから状態遷移回路のフリップフロップ部分と組み合わせ回路の切り出しを行う（ステップＳ１）。

【００２９】即ち、レジスタ（フリップフロップＦＦ０～ＦＦｎ部分）とそれを制御する組み合わせ回路部分３０により構成される状態遷移回路が図３に示すように一般的に示される。このような状態遷移回路がメモリ装置１に記憶されるネットリストから切り出される。

【００３０】この時、非同期リセット信号の有無を判断する（ステップＳ２）。非同期リセット信号がある場合は、非同期のリセット条件式を求める（ステップＳ３）。この非同期リセット信号の有無の判断及び非同期のリセット条件式を求める処理も切り出し回路２において行われる。

【００３１】次に切り出された状態遷移回路の内、フリップフロップ（ＦＦ）部分で負論理の出力ＱＮｎが着目している組み合わせ部分３０に接続している場合、出力負論理変換回路３によりＦＦの負論理出力を組み合わせ回路から取り除く処理を行う（ステップＳ４）。

【００３２】即ち、図４に示されるように、ＦＦ部分で負論理の出力ＱＮｎが着目している組み合わせ部分３０に接続している場合〔図４（１）〕、Ｑｎ にインバータ４０を挿入し、出力ＱＮｎを使用しないように変更する〔図４（２）〕。

【００３３】この段階で、図５に示すように、ＦＦ部分で負論理の出力ＱＮｎを使用しないレジスタ部分（ＦＦ０～ＦＦｎ）と組み合わせ回路３０の状態遷移回路ブロック図を得る。

【００３４】次いで、真理値表変換回路４において、組み合わせ回路３０を真理値表に変換する（ステップＳ５）。この時、真理値表変換回路４には、先に求めた非同期のリセット条件式（非同期リセット生成信号）が入力される。

【００３５】真理値表変換回路４により変換された結果は、図６に示す如くである。図６において、Ａ０で示される入力データ及び対応する現状態コードは、１または０のいずれも取りうるdon't care（無視論理）部分である。

【００３６】これを現状態コード、次状態コードのdon't care（無視論理）を展開する（ステップＳ６）。この展開された状態が図７に示される。図６のＡ０の部分が図７でＡ１、Ａ２に展開されている。

【００３７】次いで、図７に展開された真理値表に対し、状態コードに仮の状態名を割り当てる（ステップＳ７）。この仮の状態名の割り当ては、任意に設定出来るが、仮の状態名として、メモリ５に記憶されている状態名変換テーブル５に登録されているものが使用される。

【００３８】即ち、図１の状態名設定回路６において、メモリ５に記憶されている状態名変換テーブルの状態名を用いて任意に割り当てが行われる。この結果例えば、図８のように入力データ、出力データに対して仮の状態名ＳＴ０～ＳＴ３等が設定される。

【００３９】仮の状態名ＳＴ０～ＳＴ３等が設定された真理値表に対し、並べ変え回路７により、更に現状態名でソートし、現状態名毎にまとめが行われる（ステップＳ８）。このようにしてまとめられた真理値表は、図９に示される如くになる。

【００４０】したがって、この図９に対応する状態遷移図が図１０に示される。例えば、図１０において、状態ＳＴ０の時、入力データが１．．．１１であれば、次の状態は、状態ＳＴ1 となり、入力データが０．．．１－であれば、次の状態は、状態ＳＴ２となるように遷移することが理解出来る。

【００４１】このような状態遷移図に対応して、真理値表変換回路４からの真理値出力と、並べ変え回路７からの出力に基づき、論理式に変換する。

【００４２】次いで、ＨＤＬ変換回路１１において、切り出し回路２からの非同期リセット生成信号が入力されているか否かを判断し（ステップＳ９）、非同期リセット生成信号が入力されている場合は、セレクタ１０から出力される、変換ＨＤＬ選択信号９１に対応するハードウェア変換情報に基づき非同期のリセット条件式から上記変換された論理式をハードウェア記述言語変換し、生成する（ステップＳ１０）。

【００４３】更に、状態名を含んだ真理値表（図９）からハードウェア記述言語を生成して（ステップＳ１１）処理を終了する。この生成されたハードウェア記述言語は、後のＣＡＤシステムの運用のためにメモリ１２に記憶蓄積される。

【００４４】図１１は、上記のハードウェア記述言語（本例ではＶＨＤＬ）への変換の対象とした状態遷移回路の一例を示す図である。

【００４５】図１１の状態遷移回路は、フリップフロップ１１０、１１１を含み、更にこれらを制御する組み合わせ回路としてアンドゲート１１２～１１６、オアゲート１１７、１１８及びインバータ１１９～１２１を有している。

【００４６】更に、図１１の状態遷移回路の接続情報のみをリストしたネットリストが図１２に示される。そして、このネットリストを先に図１乃至図１０に基づき説明した本発明の実施例にしたがい、ハードウェア記述言語であるＶＨＤＬに変換した結果が図１３及び図１４に示される。尚、図１４は、図１３の記述に継続する記述である。

【００４７】

【発明の効果】以上実施例にしたがい説明したように本発明により、既存の回路をハードウェア記述言語に変換することができ、これにより電気的特性やテクノロジに依存しない機能シュミレーションあるいは、他のシステムへの移植性を高めることが可能となる。

【００４８】更に、ハードウェア記述言語に変換することにより、ゲート数を少なくし、また処理スピードを高める目的にあったセルにマッピングが可能である。また、状態割り付け（状態のコード付け）の変更が可能となる。

【図面の簡単な説明】

【図１】本発明の実施例構成のブロック図である。

【図２】本発明の実施例の処理フローである。

【図３】ＦＦのＱＮ 端子を含む状態遷移回路のブロックである。

【図４】ＱＮ 端子の省略を説明する図である。

【図５】図３のＱＮ 端子を削除した時の状態遷移回路のブロックである。

【図６】組み合わせ回路３０を真理値表に変換した結果を示す図である。

【図７】図６の現状態コード、次状態コードのdon't care（無視論理）を展開した真理値表である。

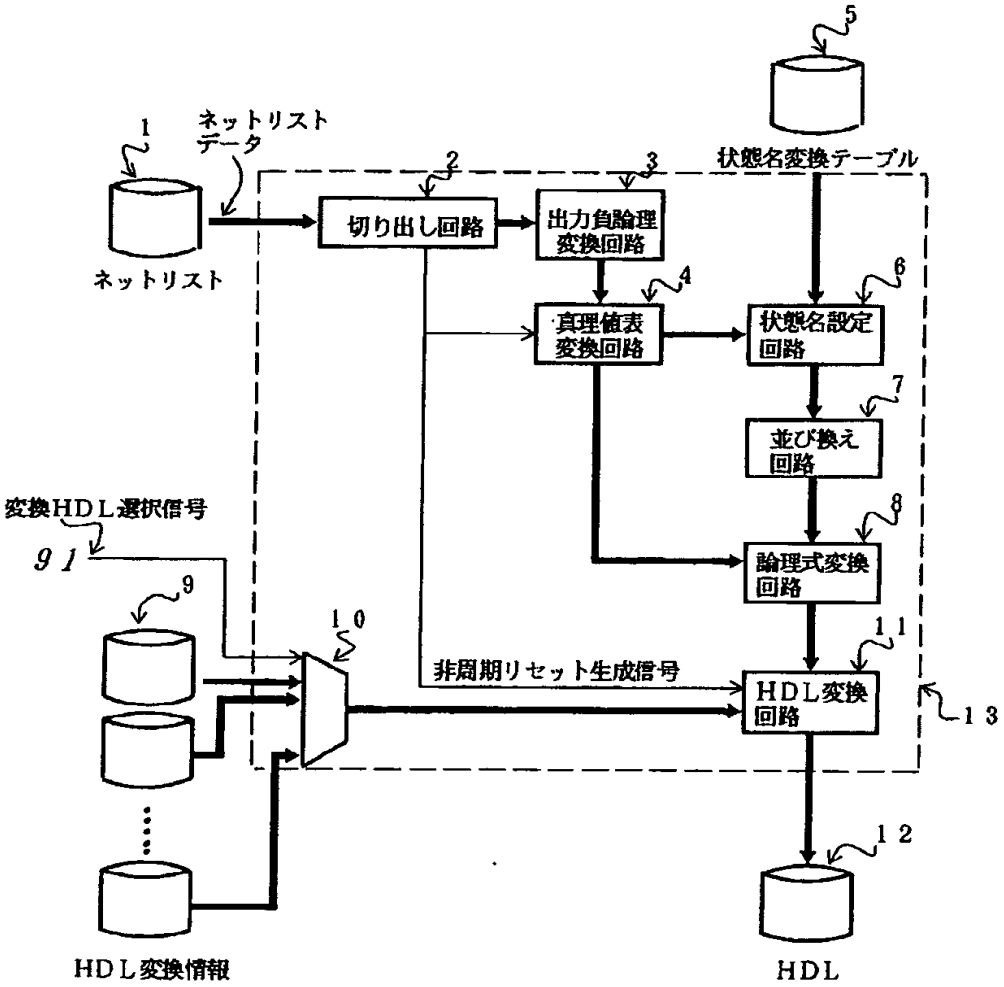
【図８】図７に現状態、次の状態に仮の状態名を設定した真理値表である。

7　　8

【図９】現状態名でソートし、現状態名毎にまとめた真理値表である。
【図１０】変換された状態遷移図である。
【図１１】状態遷移回路の一例である。
【図１２】図１１の状態遷移回路に対応するネットリストの一例である。
【図１３】変換されたハードウェア記述言語（ＶＨＤＬ）（その１）である。
【図１４】変換されたハードウェア記述言語（ＶＨＤＬ）（その２）である。
【符号の説明】
１　ネットリストを記憶するメモリ
２　切り出し回路
＊３　出力負論理変換回路
４　真理値表変換回路
５　状態名変換テーブルを記憶するメモリ
６　状態名設定回路
７　並び変え回路
８　論理式変換回路
９　ＨＤＬ変換情報
９１　変換ＨＤＬ選択信号
１０　セレクタ
１１　ＨＤＬ変換回路
１２　変換されたＨＤＬを記憶するメモリ
１３　変換処理を実行する演算処理装置

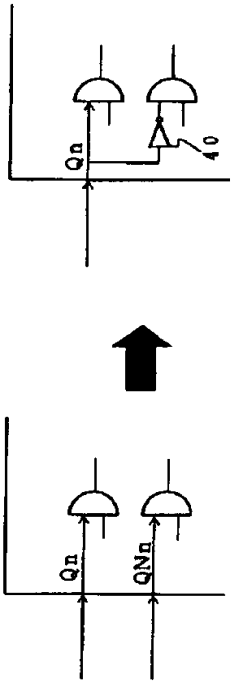
【図１】

実施例構成ブロック図



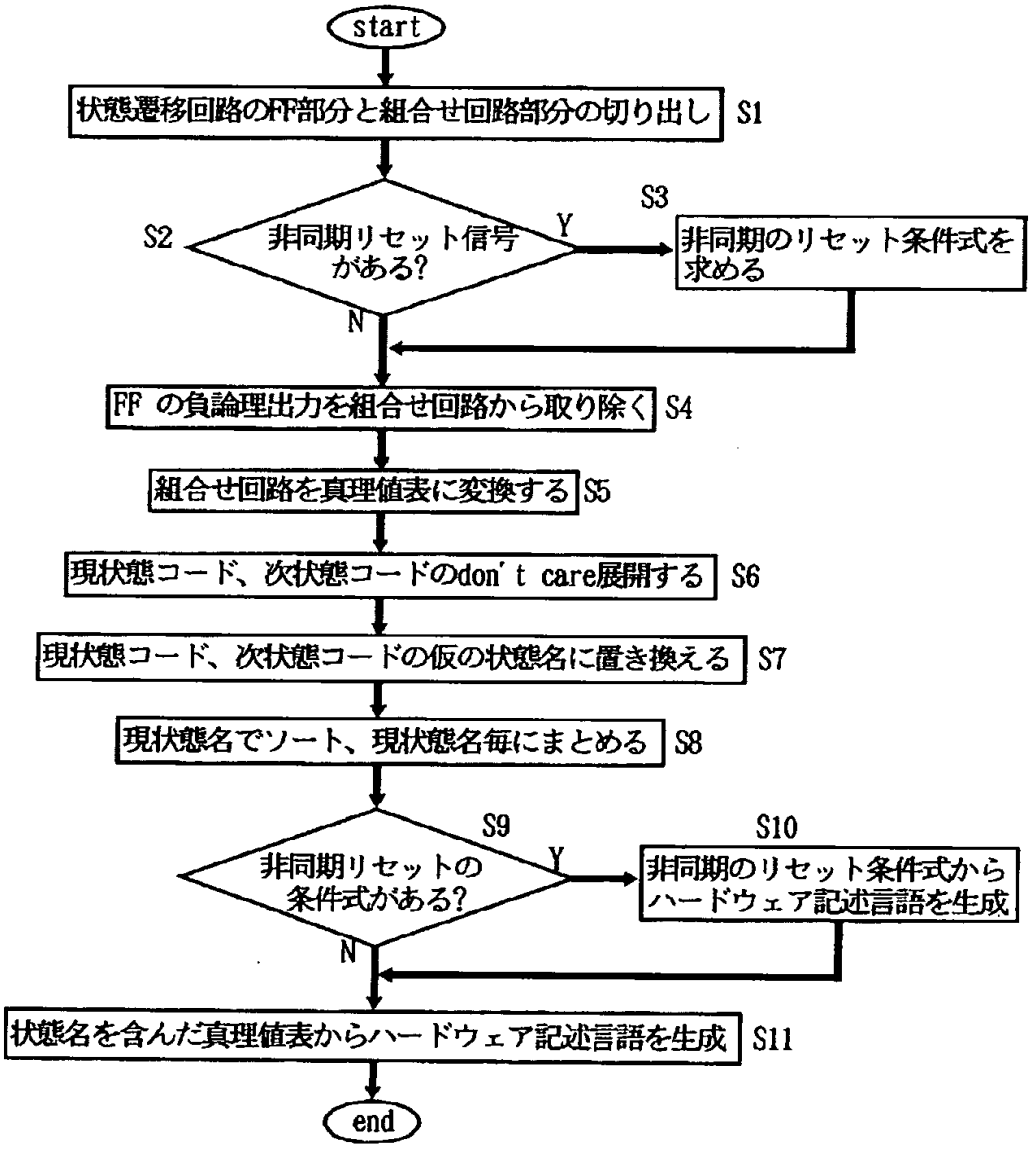
【図４】

Ｑｎ端子の省略の説明図



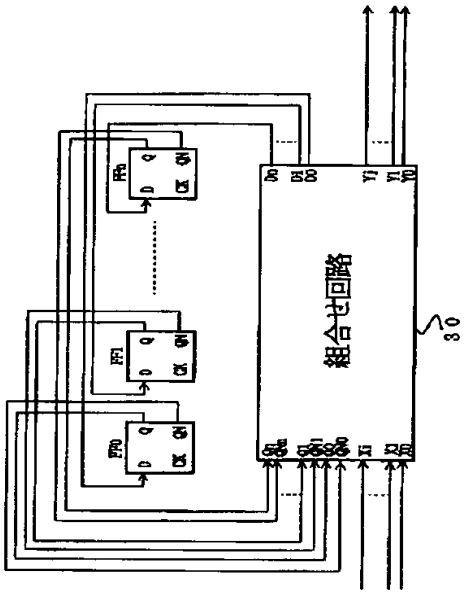
【図２】

本発明実施例処理フロー



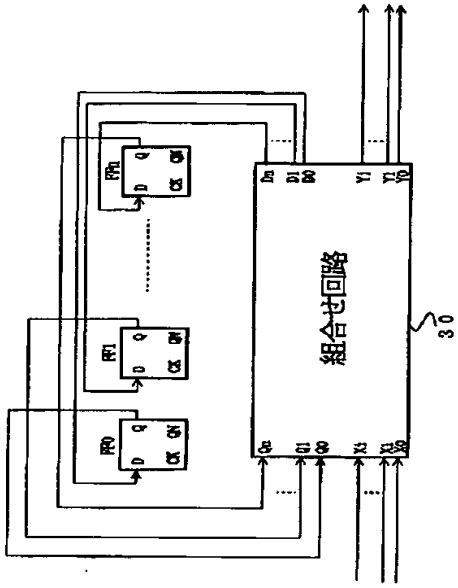
【図３】

ＦＦのQN端子を含む状態遷移回路ブロック図



【図５】

図３のQN端子を削除した時の状態遷移回路のブロック図



【図８】

図７に現状態、次の状態に仮の状態名を設定した真理値表

入力データ	現状態	次状態	出力データ
Xi…X1 X0			Yj…Y1 Y0
0… 0 1	ST1	ST3	1… 1 1
⋮	⋮	⋮	⋮
0… 1 －	ST0	ST2	1… 1 0
0… 1 －	ST1	ST2	1… 1 0
⋮	⋮	⋮	⋮
1… 1 1	ST0	ST1	1… 1 0
⋮	⋮	⋮	⋮

【図６】

組み合わせ回路３０を真理値表に変換した結果

入力データ	現状態コード	次状態コード	出力データ	
Xi…X1 X0	Qn…Q1 Q0	Dn…D1 D0	Yj…Y1 Y0	
0… 0 1	0… 1 0	1… 0 0	1… 1 1	
⋮	⋮	⋮	⋮	
0… 1 －	0… － 0	1… 0 1	1… 1 0	← A0
⋮	⋮	⋮	⋮	
1… 1 1	0… 0 0	0… 1 0	1…1 0	
⋮	⋮	⋮	⋮	

【図７】

図６の現状態コード、次状態コードのdon't careを展開した真理値表

入力データ	現状態コード	次状態コード	出力データ	
Xi…X1 X0	Qn…Q1 Q0	Dn…D1 D0	Yj…Y1 Y0	
0… 0 1	0… 1 0	1… 0 0	1… 1 1	
⋮	⋮	⋮	⋮	
0… 1 －	0… 0 0	1… 0 1	1… 1 0	← A1
0… 1 －	0… 1 0	1… 0 1	1… 1 0	← A2
⋮	⋮	⋮	⋮	
1… 1 1	0… 0 0	0… 1 0	1… 1 0	
⋮	⋮	⋮	⋮	

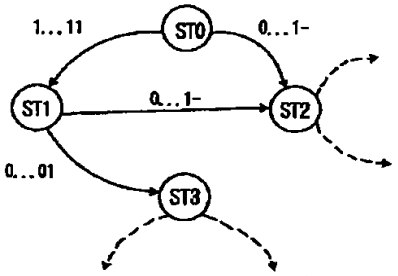
【図９】

現状態名でソートし、現状態名毎にまとめた真理値表

入力データ Xi…X1 X0	現状態	次状態	出力データ Yi…Y1 Y0
1… 1 1	ST0	ST1	1… 1 0
0… 1 -	ST0	ST2	1… 1 0
⋮	⋮	⋮	⋮
0… 1 -	ST1	ST2	1… 1 0
0… 0 1	ST1	ST3	1… 1 1
⋮	⋮	⋮	⋮

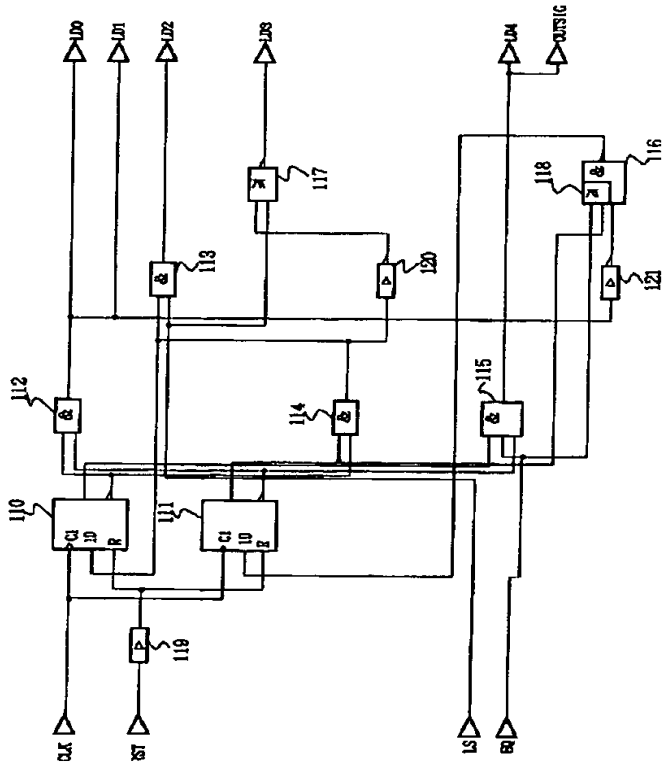
【図１０】

変換された状態遷移図



【図１１】

状態遷移回路の一例



【図１２】

図１１の状態遷移回路に対応するネットリストの一例

```
USER      : FUJITSU;
SYSTEM    : TEST;
REVISION  : 0001;
DATE      : 94/05/31;
DESIGNER  : FUJITSU;
NAME      : CNTL;
PURPOSE   : LOGSIM LAYOUT;
LEVEL     : UNKNOWN;
REVISION  : 0001;
DATA      :94/05/31;
DESIGNER  : FUJITSU;
INPUTS    : .CLK. .RST. .LS. .BQ;
OUTPUTS   : .LD0. .LD1. .LD2. .LD3. .LD4. .OUTSIG;
TYPES:
N2P : MOOHX, MOOPZ, MOOPY;
V2B : MOOGZ, MOOBZ, XXX001;
R2K : MOOGY;
N3P : MOOGX;
G23 : MOOEY;
F00 : XXX000, XXX100;
ENDTYPES:
NETS:
XXX001X : XXX001.X, XXX100.R, XXX000.R;
XXX000XQ : XXX000.XQ, MOOPZ.A2, MOOPY.A1;
MOOBZX : MOOBZ.X, MOOEY.B;
MOOGZX : MOOGZ.X, MOOGY.A1;
CLKZ : .CLK, XXX000.CK, XXX100.CK;
RSTZ : .RST, XXX001.A;
D2P_0Z : XXX000.Q, MOOGX.A1;
P2D_0Z : MOOPZ.X, MOOHX.A1, MOOGZ.A, XXX000.D;
D2P_1Z : XXX100.Q, MOOEY.A2, MOOPZ.A1;
P2D_1Z : MOOEY.X, XXX100.D;
LSZ : .LS, MOOHX.A2, MOOGY.A2;
BQZ : .BQ, MOOGX.A2, MOOEY.A1;
LD1Z : .LD1, .LD0, MOOPY.X, MOOBZ.A;
LD2Z : .LD2, MOOHX.X;
LD3Z : .LD3, MOOGY.X;
OUTSIGZ : .OUTSIG, .LD4, MOOGX.X;
NET0Z : XXX100.XQ, MOOGX.A3, MOOPY.A2;
ENDNETS;
ENDNAME;
ENDUSER;
```

【図１３】

変換されたハードウェア記述言語（VHDL）（その1）

```
entity CNTL is
  port(
     RST : in bit;
     BQ : in bit;
     LS : in bit;
     CLK : in bit;
     OUTSIG : out bit;
     LD4 : out bit;
     LD3 : out bit;
     LD2 : out bit;
     LD1 : out bit;
     LD0 : out bit
  );
end CNTL;

architecture state_transition of CNTL is
  type t_state is (ST0, ST1, ST2);
  signal state      : t_state :=ST0;
  signal next_state : t_state;
begin

   process(RST, CLK)
   begin
      if (RST='1')then
         state <= ST0;
      elsif (CLK'event and CKL = '1')then
         state <= next_state;
      end if;
   end process;

   process(BQ, LS, state)
   begin
      next_state <= state;

      case state is

        when ST0 =>
           OUTSIG <= '0';
           LD4 <= '0';
           LD3 <= '0';
           LD2 <= '0';
           LD1 <= '0';
           LD0 <= '0';

              next_state <= ST1;
```

【図１４】

変換されたハードウェア記述言語（VHDL）（その2）

```
        when ST1 =>
           OUTSIG <= '0';
           LD4 <= '0';
           LD3 <= '0';
           LD2 <= '0';
           LD1 <= '0';
           LD0 <= '0';

           if (LS) = '1' then
              next_state <= ST2;
           elsif ( not(LS) ) = '1' then
              next_state <= ST2;
           end if;

        when ST2 =>
           OUTSIG <= '0';
           LD4 <= '0';
           LD3 <= '0';
           LD2 <= '0';
           LD1 <= '0';
           LD0 <= '0';

           if ( not(BQ) ) = '1' then
              next_state <= ST1;
           elsif ( not(not(BQ)) ) = '1' then
              next_state <= ST0;
           end if;
      end case;
   end process;
end state_transition;
```